M. J. Bearden 2-4-2-2

METHOD AND APPARATUS FOR USE IN SPECIFYING AND INSURING SERVICE-LEVEL QUALITY OF SERVICE IN COMPUTER NETWORKS

Related Application

United States patent application Serial No. (M. J. Bearden-S. Garg-W. Lee-A. P. A. van Moorsel Case 3-5-3-3) was filed concurrently herewith.

Technical Field

This invention relates to computer networks and, more particularly, to monitoring and controlling quality of service (QoS) in such computer networks.

Background of the Invention

Known policy-based computer network management arrangements do not clearly distinguish the goal, i.e. the "what", of management from the policy, i.e. the "how", that achieves the goal. Indeed, most known policy-based management (PBM) arrangements require a system administrator to specify policies as declarative rules of the form "if event/condition then action." From a system administrator's viewpoint, such rules represent a "high-level" specification of "what" needs to be achieved in terms of the network/system behavior. However, from the viewpoint of a client, i.e. the end user of a service, such rules represent a "low-level" specification of "how" the client's desired system behavior is to be achieved. This makes it difficult or impossible for clients without expertise in network management to express the desired system behavior in a way supported by known PBM arrangements. Instead, a system administrator typically translates the client's "high-level" description of the desired system behavior, i.e. service-level QoS goals, into "low-level" rule-based or procedural syntax of network management policies. To summarize, in the known PBM arrangements, there is no support for specifying the client's service-level QoS goals as part of the network management policy definition.

Summary of the Invention

Problems and/or limitations of prior policy-based network management arrangements are addressed by integrating the "what" and "how" of PBM in a single framework that enables a system administrator to specify service-level QoS goals for automatic enforcement instead of, or in addition to, policy rules or procedures.

Automatic enforcement of the specified QoS goals is realized through the execution of policy logic, i.e. policy rules or procedures, without the client or system administrator having to specify the policy logic in the form of rules and/or procedures.

Specifically, one embodiment of the invention employs a management server including a graphical interface that allows a user, e.g. an administrator, to easily specify parameters for predefined types of service-level QoS goals. A QoS goal is defined by specifying a client, a service, and a QoS expression. A QoS expression is a proposition that indicates the client's desired range of values for some QoS metric, e.g. service response time or service availability. The state of the network is monitored and one or more defined QoS goals are selected for evaluation in a continuous process. The QoS delivered for the selected goal is determined and compared to the selected QoS goal. Then, prescribed actions are taken or not depending whether the delivered QoS is equal to the selected QoS goal. If not, and the delivered QoS exceeds the selected QoS goal, a set of actions is determined and executed to reduce network resources, i.e. network element resources, assigned to the client and service of the selected goal. Similarly, if the delivered QoS is worse than the selected QoS goal, a set of actions is determined and executed to increase network resources assigned to the client and service of the selected goal.

In an embodiment of the invention, QoS goals are stored in a goal repository and continuously updated by adding, redefining, or removing a service-level QoS goal as requested by an administrator.

Similarly, in an embodiment of the invention, the state of the network is monitored and the results are stored in a monitored state repository. Also stored in the monitored state repository are state changes indicated by received network event notifications.

Brief Description of the Drawing

FIG. 1 is a table illustrating in simplified and generalized form example "high-level" QoS management goals;

FIG. 2 is a table illustrating in simplified form an example policy procedural logic;

FIG. 3 shows, in simplified form, details of a network employing an embodiment of the invention; and

FIG. 4 is a flow chart illustrating steps in a process employed in an embodiment of the invention.

Detailed Description

Again, it is noted that current PBM arrangements require system administrators to specify rules that to the system administrator represent the specification of "what" needs to be achieved in terms of network/system behavior. From a client viewpoint, however, these rules represent the "how" of management. The client is really interested in service-level QoS goals such as "my transaction failure rate should be less than some specific value" or "my end-to-end response should be less than some specific value". Indeed, there is currently no support in prior arrangements for specifying the client's service-level QoS goals as part of the management policy definition.

FIG. 1 is a table illustrating in simplified form example QoS goals. As shown, QoS goals are represented using the generalized goal template shown in TABLE 1, namely, "during T, satisfy Q for client C that uses service S". The goal parameters, in this example, are defined as follows. Parameters C and S identify respectively a client and some service accessed by the client, such as a Web or DNS server, a networked application server, or a file server. Parameter Q is a QoS expression with three parts, as follows. The first part is Q.metric that identifies a QoS metric such as end-to-end service delay, transaction failure rate, etc.; the second part is Q.op, an operator used to compute whether the client's delivered QoS value satisfies the desired QoS; the third part is Q.value, a value that represents the desired QoS for the given QoS metric. The QoS expression will evaluate to "true" or "false" at run time when the given operator Q.op is used to compare the delivered QoS for the given metric to the desired QoS, Q.value. Finally, parameter T identifies a time range when the given QoS goal is intended to have effect. By way of an example, consider an end-to-end QoS goal as follows: "Provide client Joe with average SAP transaction delay of at most 1 second," where SAP is an example networked service. This QoS goal is represented by setting the parameters of TABLE 1 as follows: C = "Joe", S = "SAP", Q.metric = "AvgSAPTransactResponse", Q.op = "≤", Q.value = "1 second", and T = "Always". The goal is satisfied when client

Joe's average SAP transaction response time is determined, through observation and/or estimation, to be less than or equal to one second. The goal is not satisfied whenever Joe's SAP average transaction response time is determined to be greater than one second.

FIG. 2 is a table illustrating in simplified form an example policy procedural logic. Specifically, shown in TABLE 2 is pseudocode for one possible (simplistic) procedure for enforcing the above QoS goal in a networked system with priority-based packet switching and a function defined as getClientQoS() that measures or computes a client's transaction delay. The example pseudocode is explained as follows. The "if" condition in line 1 is satisfied when the delivered QoS for client C using service S does not satisfy the QoS expression of the goal specified for client C and service S. Line 3 specifies an example action that is expected to help the delivered QoS for client C to achieve the value specified in the goal. Specifically, in this example the priority for network traffic is increased, for traffic generated by client C accessing service S. Lines 5 through 10 specify a rule of the form "if condition then action" similar to those commonly supported by existing PBM arrangements and by existing QoS-enabled network devices. In lines 5 through 7, the condition part of the rule identifies that a given packet P is part of client C's communication with service S. Line 9 contains the action part of the rule, which in the given example is setting the appropriate priority for switching and queuing the packet at network elements. Line 4 indicates that this policy rule should be enforced at each element in the network over which client C accesses service S.

It is noted that such a procedure can be specified by a management expert in advance and reused for a number of different goal parameters, i.e. for different clients, services, QoS metrics, etc. Indeed, the procedural policy specification is highly dependent on the types of parameters assigned to the QoS goal template, and on the types of elements in the networked system that can be controlled in order to enforce QoS.

It is felt best to consider some terminology. Thus, a "policy" is defined to be a process that implements a function with a parameter called an "objective." An "objective" is defined to be a Boolean expression over some set of goals. A goal is a proposition defined as above on (1) a "service", i.e. an application, (2) a "client" that accesses the service, (3) a "time range" for goal enforcement, and (4) a "QoS

"expression" specified using applicable metric identifiers, operators, and values. Again, an example goal proposition is as follows: Client = Joe, Service = SAP, Time = Always and TransactionDelay ≤ 1 ms. A "policy instance" P(H) exists whenever objective H is enforced using policy P. The inputs to a policy instance are state updates of the client, service, and network elements that allow the client to access the service, and its outputs are a combination of (1) policy rules consumed by rule-based PBM software, (2) control signals sent to network elements and resources to affect the client's QoS, and/or (3) notifications sent to a administrative interface, including but not limited to service-level alarms and suggested manual network provisioning actions.

An embodiment of the invention allows the user (system administrator) to specify a set of goals and objectives in a goal repository, describing the "what" of the service-level QoS management. For each goal G in some objective H specified, a policy is automatically selected such that the effect of executing P(G) is to enforce goal G to the extent that is feasible given the available networked resources. Thus, the invention accepts only a limited set of goals for which the embodiment of the invention contains the policy logic needed to enforce those goals. The "how" of enforcing the goals is specified by the policy logic contained in the invention's embodiment. As noted earlier, a simplified example of policy logic is given in TABLE 2 of FIG. 2.

In this embodiment of the invention there is an explicit distinction between the "what" and "how" of PBM and each one is an integral part of a policy framework and a PBM solution which enables the following:

- Directly specifying (and modifying on-line) clients' service-level QoS goals as part of the network management objective;
- Easily reporting and verifying the effectiveness of policy-based management for achieving these goals;
- Providing feedback so that policy logic can be modified, either manually off-line or automatically on-line to achieve desired QoS goals;
- Enabling service providers and clients to establish service-level agreements (SLA's) based on goals and objectives that are mutually understood. As an integral part of the policy specification, such SLA's enable an SLA-based

revenue model for service providers, as opposed to the flat-rate pricing that is presently the norm for service providers.

Sub B1

~~FIG. 3 shows, in simplified form, details of network 300 employing an~~ embodiment of the invention. Specifically, shown are management server 301 including an embodiment of the invention, an associated graphical user interface 302, goal repository 304, and monitored state repository 305. Management server 301 is controllably connected to a data communication network 306, for example, the internet or World Wide Web (the Web), that includes a set of one or more configurable QoS-enabled network elements 307-1 through 307-N. A set of service servers 308-1 through 308-X is also controllably connected to data communications network 306. Finally, a set of client stations 309-1 through 309-Y is also controllably connected to data communications network 306. It is noted that client stations 307 may each be a personal computer (PC), workstation or the like for accessing data communication network 306, i.e. the internet. In this example, configurable QoS-enabled network elements 307 may include network routers and switches, network traffic shapers, application-level traffic redirectors, and application-level or network-level load balancers; the service servers 308 may include file servers (e.g. NFS), database servers (e.g. SQL), naming servers (e.g. DNS), network directories (e.g. LDAP), enterprise resource planning software (e.g. SAP or PeopleSoft), or servers running any other networked application; and the clients 309 may include thin client terminals, personal digital assistants, telephony or video devices, or web browsers, ~~applets, agents, or client programs running on personal computers or workstations.~~

FIG. 4 is a flow chart illustrating steps in a process employed in an embodiment of the invention. The process is started in step 401. At run time, a system administrator or the like employing user interface 302 defines a service level QoS goal by selecting a client from 309-1 through 309-Y, an application from service servers 308-1 through 308-X, and a QoS expression. Stated another way, user interface 302 allows the system administrator to specify goal parameters for an objective "H" and choose when attempted enforcement of "H" should begin. At that time management server 301 parameterizes and instantiates P(G) for each goal G that is part of objective H, such that the policy logic of P is appropriate for enforcing goal G. For example, the policy logic described in simplified form in FIG. 2 could be selected to enforce a goal of the form given in FIG. 1.

Thereafter, in this example, three processes run concurrently in management server 301. Specifically, a first sub-process includes maintaining the goal repository 304, a second sub-process maintains the monitored state repository 305 and a third sub-process effects the QoS management of defined goals. Again, these three sub-processes, once started, run concurrently and continuously.

The first sub-process of maintaining the goal repository 304 includes step 402 that causes the updating of goal repository 304 by adding, redefining or removing a service level QoS goal. Then, step 403 tests to determine whether the contents of goal repository 304 should be modified. If the test result in step 403 is YES, control is returned to step 402 that causes the updating of goal repository 304. If the test result in step 403 is NO, the test is iterated until a YES result is obtained and control is again returned to step 402. Steps 402 and 403 are continuously iterated, as described above.

The second sub-process of maintaining the monitored state repository 305 includes step 404 that tests to determine whether a network state update should be requested now. The network update is realized by monitoring elements employed in the network, including both clients and service servers. In one example, the monitoring is realized by obtaining measures of service performance statistics. The desired measurements are collected, in one example, by employing a so-called VitalAgent client-side monitoring software commercially available from Lucent Technologies Inc. See for example, http://www.ins.com/software/ for VitalAgent product documentation. If the test result in step 404 is YES, step 405 causes a query of the network element(s). Thereafter, step 407 tests to determine whether a network event notification has been received either for previously requested network state update or for an (unrequested) network update, i.e. an asynchronous or trap update. Such an event notification indicates that the state of the network has changed by the event that has occurred. If the test result in step 407 is YES, then step 406 causes the updated state to be stored in monitored state repository 305. Otherwise, if the test result in step 407 is NO, the control is returned to step 404. Returning to step 404, if the test result is NO, no update is requested and control is passed to step 407. Appropriate ones of steps 404 through 407 are iterated continuously.

The third sub-process of effecting the desired QoS management of defined goals includes step 408 that sets a counter X equal to one (1). Then step 409 causes the

selection of the X^{th} defined QoS goal in goal repository 304. Step 410 causes the use of the contents of monitored state repository 305 to determine the delivered QoS for the selected goal. In step 411, the delivered QoS is compared with the QoS expression of the selected QoS goal. Then, step 412 tests to determine if the delivered QoS is equal to the desired QoS for the selected QoS goal. If the test result in step 412 is YES, control is passed to step 413. Step 413 causes a report to be sent to the system administrator indicating that the delivered QoS satisfies the selected QoS goal. Then, step 414 causes counter X to be incremented by one (1), namely, X = X+1. Step 415 tests to determine if X in counter X is greater than the number of QoS goals in goal repository 304. If the test result in step 415 is YES, control is returned to step 408 and steps 408 through 415 are iterated until either step 412 or step 415 yields a NO result. If step 415 yields a NO result, control is returned to step 409 and steps 409 though 415 are iterated until step 412 yields a no result or step 415 yields a YES result. If step 412 yields a NO result, control is passed to step 416. Step 416 tests to determine whether the delivered QoS exceeds the selected QoS goal. If the test result in step 416 is YES, control is passed to step 417. Step 417 determines and executes a set of actions to reduce network resources, i.e. the resources of network elements such as buffer space or reserved bandwidth at a router, assigned to the client and service of the selected QoS goal. Thereafter, control is passed to step 413 that reports the set of actions taken in step 417, and the results of said actions, to the system administrator. Returning to step 416, is the test result is NO, control is passed to step 418. Step 418 determines and executes a set of actions to increase the amount of network resources assigned to the client and service of the selected QoS goal. Thereafter, control is passed to step 413 that reports the set of actions taken in step 417, and the results of said actions, to the system administrator. Then, control is passed to step 414 that increments counter X by one (1). Thereafter, step 415 is repeated as described above, and appropriate ones of steps 408 through 418 are iterated continuously. It is noted that the computations performed in the control flow, especially in steps 404, 410, 417 and 418, must vary depending upon the type of parameters included in the enforced goals and with the types of elements in the networked system wherein the QoS goals are enforced.

The above-described embodiments are, of course, merely illustrative of the principles of the invention. Indeed, numerous other methods or apparatus may be devised by those skilled in the art without departing from the spirit and scope of the invention.